Table 1 (cont')

```
        for (nn = nm = 0, more = 1; more; ) {                          ...pr_align
                for (i = more = 0; i < 2; i++) {
                        /*
 5                       * do we have more of this sequence?
                         */
                        if (!*ps[i])
                                continue;

10                      more++;

                        if (pp[i].spc) {        /* leading space */
                                *po[i]++ = ' ';
                                pp[i].spc--;
15                      }
                        else if (siz[i]) {      /* in a gap */
                                *po[i]++ = '-';
                                siz[i]--;
                        }
20                      else {                  /* we're putting a seq element
                                                 */
                                *po[i] = *ps[i];
                                if (islower(*ps[i]))
                                        *ps[i] = toupper(*ps[i]);
25                              po[i]++;
                                ps[i]++;

                                /*
                                 * are we at next gap for this seq?
30                               */
                                if (ni[i] == pp[i].x[ij[i]]) {
                                        /*
                                         * we need to merge all gaps
                                         * at this location
35                                       */
                                        siz[i] = pp[i].n[ij[i]++];
                                        while (ni[i] == pp[i].x[ij[i]])
                                                siz[i] += pp[i].n[ij[i]++];
                                }
40                              ni[i]++;
                        }
                }
                if (++nn == olen || !more && nn) {
                        dumpblock();
45                      for (i = 0; i < 2; i++)
                                po[i] = out[i];
                        nn = 0;
                }
        }
50  }

    /*
     * dump a block of lines, including numbers, stars: pr_align()
     */
55  static
    dumpblock()                                                        dumpblock
    {
        register  i;

60      for (i = 0; i < 2; i++)
                *po[i]-- = '\0';
```

Table 1 (cont')

...dumpblock

```
        (void) putc('\n', fx);
        for (i = 0; i < 2; i++) {
            if (*out[i] && (*out[i] != ' ' || *(po[i]) != ' ')) {
                if (i == 0)
                    nums(i);
                if (i == 0 && *out[1])
                    stars();
                putline(i);
                if (i == 0 && *out[1])
                    fprintf(fx, star);
                if (i == 1)
                    nums(i);
            }
        }
}

/*
 * put out a number line: dumpblock()
 */
static
nums(ix)                                                    nums
        int     ix;     /* index in out[] holding seq line */
{
        char            nline[P_LINE];
        register        i, j;
        register char   *pn, *px, *py;

        for (pn = nline, i = 0; i < lmax+P_SPC; i++, pn++)
                *pn = ' ';
        for (i = nc[ix], py = out[ix]; *py; py++, pn++) {
                if (*py == ' ' || *py == '-')
                        *pn = ' ';
                else {
                        if (i%10 == 0 || (i == 1 && nc[ix] != 1)) {
                                j = (i < 0)? -i : i;
                                for (px = pn; j; j /= 10, px--)
                                        *px = j%10 + '0';
                                if (i < 0)
                                        *px = '-';
                        }
                        else
                                *pn = ' ';
                        i++;
                }
        }
        *pn = '\0';
        nc[ix] = i;
        for (pn = nline; *pn; pn++)
                (void) putc(*pn, fx);
        (void) putc('\n', fx);
}

/*
 * put out a line (name, [num], seq, [num]): dumpblock()
 */
static
putline(ix)                                                 putline
        int     ix;
{
```

Table 1 (cont')

...putline

```
        int             i;
        register char   *px;

        for (px = namex[ix], i = 0; *px && *px != ':'; px++, i++)
                (void) putc(*px, fx);
        for (; i < lmax+P_SPC; i++)
                (void) putc(' ', fx);

        /* these count from 1:
         * ni[] is current element (from 1)
         * nc[] is number at start of current line
         */
        for (px = out[ix]; *px; px++)
                (void) putc(*px&0x7F, fx);
        (void) putc('\n', fx);
}

/*
 * put a line of stars (seqs always in out[0], out[1]): dumpblock()
 */
static
stars()                                                          stars
{
        int             i;
        register char   *p0, *p1, cx, *px;

        if (!*out[0] || (*out[0] == ' ' && *(po[0]) == ' ') ||
           !*out[1] || (*out[1] == ' ' && *(po[1]) == ' '))
                return;
        px = star;
        for (i = lmax+P_SPC; i; i--)
                *px++ = ' ';

        for (p0 = out[0], p1 = out[1]; *p0 && *p1; p0++, p1++) {
                if (isalpha(*p0) && isalpha(*p1)) {

                        if (xbm[*p0-'A']&xbm[*p1-'A']) {
                                cx = '*';
                                nm++;
                        }
                        else if (!dna && _day[*p0-'A'][*p1-'A'] > 0)
                                cx = '.';
                        else
                                cx = ' ';
                }
                else
                        cx = ' ';
                *px++ = cx;
        }
        *px++ = '\n';
        *px = '\0';
}
```